03-23-00

A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

UTILITY PATENT APPLICATION TRANSMITTAL
UNDER 37 CFR 1.53(b)

Address to:	Attorney Docket No.	AM9-99-0247
Assistant Commissioner for Patents	Inventor(s)	M. Carey et al.
Box Patent Application	Express Mail Label No.	EK619175305US
Washington, DC 20231	Total Pages	48

Title of Application: USING AN XML QUERY LANGUAGE TO PUBLISH RELATIONAL DATA AS XML

Transmitted with the patent application are the following:

2 Page(s) Transmittal form (and one copy)
28 Page(s) Specification, claims, abstract
9 Page(s) Informal Drawings
3 Page(s) Declaration and Power of Attorney
6 Page(s) Assignment and Recordation Form (and one copy)

Fee Calculation

	Claims			Extra	Fees
Basic Fee					$ 690.00
Total Claims	44	-20 =	24	× $18.00	$ 432.00
Independent Claims	4	- 3 =	1	× $78.00	$ 78.00
Multiple Dependent				+ $260.00	$ 0.00
				Assignment	$ 40.00
				TOTAL	$1240.00

The Commissioner is hereby authorized to credit overpayments or charge fees required under 37 CFR 1.16 or 1.17 to Deposit Account 09-0441. A duplicate copy of this transmittal is attached.

EXPRESS MAIL CERTIFICATE

I hereby certify that the above paper/fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated below and is addressed to the Assistant Commissioner for Patents, Washington, DC 20231

Date of Deposit: March 21, 2000

Person mailing paper/fee: Marc D. McSwain

Signature Marc D. McSwain

Respectfully submitted,

Marc D. McSwain

Marc D. McSwain
Reg. No. 44,929
Agent for Applicants
Telephone (408) 927-3364
IBM Corporation
650 Harry Road (C4TA/J2B)
San Jose, CA 95120-6099

Using an XML Query Language to Publish Relational Data as XML

BACKGROUND OF THE INVENTION

1. Field of the Invention.

This invention relates in general to accessing databases on a network, and more specifically, to translating eXtensible Markup Language (XML) queries into queries against a relational database for the purpose of publishing relational data as XML.

2. Description of Related Art.

Computer systems incorporating Relational Database Management System (RDBMS) software using the Structured Query Language (SQL) interface are well known in the art. The SQL interface has evolved into a standard language for RDBMS software and has been adopted as such by both the American Nationals Standard Organization (ANSI) and the International Standards Organization (ISO).

In an RDBMS, all data is structured as tables. The SQL interface allows users to formulate relational operations on the tables either interactively, in batch files, or embedded in host languages such as C, COBOL, etc. Operators are provided in SQL that allow the user to manipulate the data, wherein each operator operates on either one or two tables and produces a new table as a result. The power of SQL lies in its ability to join and relate information from multiple tables or views, and the ability to perform complex operations with a single statement.

An alternative data format to the tables found in an RDBMS is XML, which is a tag language for describing documents. XML is a restricted form of SGML, the

Standard Generalized Markup Language defined in ISO 8879. XML is more fully described at the URL:

http://www.w3.org/TR/1998/REC-xml-19980210

An XML document has two parts: the marked up document, and the document schema. XML documents are made up of storage units called elements, which can be nested to form a hierarchical structure. Elements can contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure.

XML schemas specify constraints on the structures and types of elements in an XML document. The basic schema for XML is the DTD (Document Type Definition). Other XML schema definitions are also being developed, such as XML Schema, which is described at the URL:

http://www.w3.org/TR/1999/WD-xmlschema-1-19990924

Several query languages have been proposed for searching and reformatting XML documents. One such language is XML-QL, which is described at the URL:

http://www.w3.org/TR/NOTE-xml-ql

In XML-QL, queries have the general form:

```
WHERE <pattern>
 IN <URLs of input documents>
CONSTRUCT <output XML>
```

The input documents are searched for the pattern in the WHERE part of the query. Data bindings, which are specified in the pattern, are used to construct the output XML. Note that the output XML does not have to be a well-formed document with matching start and end tags, nor does it have to conform to any DTD. Important operational features of XML-QL include:

- Support for navigation in the search pattern of a query. This navigation can be expressed using path expressions.
- The ability to join and relate information from multiple input documents in a single query.
- Support for nested subqueries in the CONSTRUCT part of a query.
- The ability to generate output XML with a different structure than the input documents.

Other XML query languages that have been proposed generally support only a subset of the above operational features. For example, the XML Query Language (XQL) does not support joins, subqueries, or the ability to generate output with a different structure than its input document. XQL is described at the URL:

http://www.w3.org/tandS/QL/QL98/pp/xql.html

XML is likely to become a future standard for information exchange between peer data stores, and between client visualization tools and data servers. Most business data will probably continue to be stored in relational database systems, however. Thus, there is a need for a mechanism to publish and query relational data as XML.

SUMMARY OF THE INVENTION

To overcome the limitations in the prior art described above, and to overcome other limitations that will become apparent upon reading and understanding the present specification, the present invention discloses a method, apparatus, and computer program product to translate XML queries into queries against a relational database to publish relational data as XML. Each relational table is mapped to a virtual XML document, and XML queries are issued over these virtual documents. An XML query is parsed and transformed into a language-neutral intermediate representation, which is a sequence of operations describing how the output document is derived from the underlying relational tables. The intermediate representation is then translated into an SQL query over the underlying relational tables and into instructions for a tagger. The SQL query is executed, and the SQL query results are then fed into the tagger, which follows its instructions to generate the marked up output.

BRIEF DESCRIPTION OF THE DRAWINGS

Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. 1 is a schematic diagram of the hardware environment of the preferred embodiment of the present invention;

FIG. 2 is a block diagram depicting the Xperanto system of the preferred embodiment of the present invention.

FIG. 3 depicts the mapping of a relational table to its Default View in XML according to the preferred embodiment of the present invention.

FIG. 4 is an exemplary XML query over the Default View and its output.

FIG. 5 illustrates the Intermediate Representation of the Default View according to the preferred embodiment of the present invention.

FIG. 6 illustrates the Intermediate Representation of a query over the Default View according to the preferred embodiment of the present invention.

FIG. 7 illustrates the rewritten form of the Intermediate Representation according to the preferred embodiment of the present invention.

FIG. 8 illustrates the translation of the Intermediate Representation to a query over the relational database according to the preferred embodiment of the present invention.

FIG. 9 is a flowchart that illustrates the method steps performed by the preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

OVERVIEW

The preferred embodiment of the present invention describes a distributed computing architecture for use on the World Wide Web (WWW). Specifically, a server uses a relational database system as its data store. A mapping is established from each table in the database to a virtual XML document. Clients (or other servers) query these virtual documents using XML-QL. An XML-QL query is transformed into a language-neutral intermediate representation. The intermediate representation is then translated into an SQL query over the underlying relational table, and into tagging instructions. The SQL query is executed and the SQL query results are then fed into the tagger, which follows the tagging instructions to generate the marked up output from the SQL query results. Those skilled in the art will recognize that another XML query language could be used in place of XML-QL without departing from the scope of the present invention.

HARDWARE ENVIRONMENT

FIG. 1 schematically illustrates an exemplary computer hardware environment that could be used with the present invention, and more particularly, illustrates a typical distributed computing architecture using a network 120 to connect clients 130 and servers 110. Clients 130, which may include Web browsers 132, connect to servers 110, which execute server software such as Web daemons 106. Server-to-server

connections may also be established using Web daemons 106. A typical combination of resources may include clients 130 that are personal computers, or workstations, and servers 110 that are personal computers, workstations, minicomputers, or mainframes. These systems may be coupled to one another over network 120, which may include other networks such as LANs, WANs, SNA networks, as well as the Internet.

The preferred embodiment of the present invention comprises a system known as Xperanto 104, which maps each table in RDBMS 102 (describing contents of a database 100) to a virtual XML document. Queries in XML-QL are issued over these virtual documents. Xperanto 104 translates an XML-QL query to an SQL query over the underlying relational tables. The SQL query is executed against RDBMS 102, and the results of the SQL query are then marked up by Xperanto 104 to form the desired output XML.

In general, Xperanto 104 comprises data and instructions which, when read, interpreted, and executed by server 110, cause server 110 to perform the steps of the present invention. Generally, the data and instructions are embodied in and readable from a device, carrier or media, such as memory, data storage devices, and remote devices coupled to server 110 via a data communications device (not shown).

However, those skilled in the art will recognize that the exemplary environment and components illustrated in FIG. 1 are not intended to limit the present invention. Indeed, those skilled in the art will recognize that other alternative hardware environments may be used without departing from the scope of the present invention.

Thus, the present invention may be implemented as a method, apparatus, or article of manufacture using standard programming and engineering techniques to produce software, hardware, firmware, or any combination thereof. In addition, the term "article of manufacture" as used herein is intended to encompass logic and data embodied in or accessible from any device, carrier, or media.

OPERATION OF THE INVENTION

FIG. 2 is a block diagram that illustrates the components of Xperanto system 104. In the preferred embodiment, Xperanto system 104 is comprised of five major components: a schema mapper 200, a parser 210, a rewrite engine 220, a translator 230, and a tagger 240.

Schema mapper 200 establishes a default XML schema (the Default View) for the data stored in RDBMS 102. Each table is mapped to a virtual XML document over which XML queries are issued. These are virtual XML documents in the sense that they are preferably never materialized. Only the results of XML queries are materialized.

Parser 210 converts an XML query to a language-neutral intermediate representation. The intermediate representation is a sequence of operations that describe how the marked up output of the XML query is derived from the underlying relational tables. It includes operations to construct new XML elements from data bindings.

Rewrite engine 220 takes the intermediate representation generated by parser and converts it to an equivalent form that may be easily translated to an SQL query.

Translator 230 translates the rewritten intermediate representation into:

(1) an SQL query over the underlying relational tables and

(2) instructions for tagger 240.

Tagger instructions describe how to generate the marked up output from the SQL query results. The SQL query is executed, and the SQL query results are fed into tagger 240, which then generates marked up output according to its instructions.

Using the above components, Xperanto 104 provides the means to establish a mapping from a relational schema to an XML schema and the means to issue XML queries against that XML schema for the purpose of publishing relational data as XML.

As its name implies, the Default View provides a default XML view of the relational data. In Xperanto 104, alternative views of the data may be manually defined using XML queries, since each query effectively defines a different view of the data. XML queries may then be issued over these alternative views and the Default View. Note that views may be defined on top of views by recursive application of this procedure.

The Default View

FIG. 3 provides an exemplary Default View for a department and employee database, consisting of a DEPT table 300 and an EMP table 310. The DNO and ENO

columns in tables 300 and 310 uniquely identify the department number and employee number, respectively. As shown, the table name (either DEPT or EMP) is used as the tag for the outer-most XML element in the Default View. Each row in a table is mapped to a ROW element. Within a ROW element, each column of the table is mapped to an element with the column's name as its tag. Those skilled in the art will recognize that another schema mapping could be used in the Default View without departing from the scope of the present invention.

A Query Over the Default View

FIG. 4 provides an exemplary XML-QL query over the Default View of FIG. 3. The query provides the names of departments with more than 75 employees. The WHERE part of the query specifies the search criteria, namely that only departments with more then 75 employees are to be selected. It also specifies data bindings for DNO, NAME, and BUDGET. The CONSTRUCT part of the query specifies the structure of the output XML. As shown, a nested subquery appears within the CONSTRUCT. The subquery is used to nest employees with their departments. The binding for DNO is used to relate departments and employees in the subquery.

The Intermediate Representation

The intermediate representation is a sequence of parameterized operations that describe how the output of the XML-QL query is derived from the underlying relational

tables. The input of each operation is one or more tables, and the output is another table. Each row of a table has a number of columns, where a column is either an atomic data value, such as an integer, or XML.

Various operations in the intermediate representation are described below. Each operation's name is given along with a list of input parameters. The list of input parameters is not necessarily exhaustive. Only key parameters are described:

- BIND(X, B): This operation binds variables to data values. It produces an output table consisting of all the possible data bindings B for the input document X.
- SELECT (S): This operation selects data. It applies S to each row of its input table and produces an output table consisting of rows that satisfy S.
- CONSTRUCT(T, O): This operation constructs new XML elements. For each row of the input table, the data bindings in the XML template T are instantiated. The resulting XML element is assigned to column O of the output table.
- JOIN(R): This operation has the same semantics as a join in an RDBMS. R is a predicate that is used to relate the two input tables.
- GROUP(C, X, O): This operation groups related XML elements. The input table is grouped on column(s) C. Within a group, column X of the input table is concatenated and assigned to column O in the output. If C is empty, one group is created for the whole input table.
- NEST(R, X, O): This operation nests XML elements in a parent-child relationship. It takes two input tables, with one playing the role of the parent and the other

playing the role of the child. R is a predicate that used to relate a parent and child. Children are grouped with their parent. Within a group, column X of the child table is concatenated and assigned to column O in the output.

Those skilled in the art will recognize that another intermediate representation with different operations could be used without departing from the scope of the present invention. In general, operations are needed to establish the selection criteria to be applied to the input documents, the required relationships between the input documents, and the structure of the output XML.

FIG. 5 shows the intermediate representation for the Default View of DEPT table 300 in FIG. 3. Working from the bottom up, DEPT table 300 is accessed and input to a CONSTRUCT operation 510. For each row, data bindings from the DNO, NAME, and BUDGET columns are used to construct a ROW element, which is assigned to the RowEle column. ROW elements are then concatenated by a GROUP operation 520 and assigned to the Rows column. Finally, a second CONSTRUCT operation 530 adds the DEPT tag, and the resulting XML is assigned to the DefView column. The intermediate representation for the Default View of the EMP table would look similar.

FIG. 6 shows the intermediate representation for the query in FIG. 4. As shown, the starting point is the Default View of the DEPT and EMP tables 300 and 310. BIND operations 600 and 610 map the Default View back into a table. The marked up output is constructed by operations 630, 640, and 650. CONSTRUCT operation 630 constructs

EMP elements, and NEST operation 640 groups EMP elements with the same DNO. Finally, CONSTRUCT operation 650 generates a BIGDEPT element for each department with size greater than 100, as selected by SELECT operation 620.

The Rewrite Engine

Rewrite engine 220 takes the intermediate representation generated by parser 210 and converts it to an equivalent form that may be easily translated to an SQL query. More specifically, operations to construct and group XML elements are reordered to appear last in the rewritten intermediate representation. The steps to carry out the rewrite process are as follows:

1) Let S denote the sequence of CONSTRUCT, GROUP, and CONSTRUCT operations following a table access for the Default View of a table T. Whenever S is followed by a BIND operation B, eliminate both S and B, leaving just the table access for T.

2) Let N denote a NEST operation and S denote any sequence of CONSTRUCT and GROUP operations for the child input of N. N is replaced by a JOIN operation, followed by S and a new GROUP operation G, which performs the child grouping that was previously done by N.

FIG. 7 shows the rewritten version of the intermediate representation in FIG. 6. As shown, the operations to construct and group XML elements now appear last. FIG. 7

is for a query with only single level of nesting. However, deeper levels of nesting can be handled by repeated application of steps 1) and 2).

Those skilled in the art will recognize that different rewrite steps could be used without departing from the scope of the present invention. In general, rewrite engine 220 converts the intermediate representation to an equivalent form where operations to generate the structure of the output XML are performed last.

Translation

Translator 230 takes the intermediate representation produced by rewrite engine 220 and translates it to: (1) an SQL query over the underlying relational tables; and (2) tagger instructions. More specifically, operations on the intermediate representation are partitioned into two groups: operations that select and relate data 730, and operations that construct and group XML elements 740. Operations that select and relate data 730 are translated into an SQL query, while operations that construct and group XML elements 740 (including CONSTRUCT operation 700, GROUP operation 710, and CONSTRUCT operation 720) are translated into tagger instructions.

FIG. 8 shows the translation of the intermediate representation into an SQL query 800 and tagger instructions 810. A straightforward translation from the intermediate representation into SQL query 800 is shown. Those skilled in the art will recognize that a different SQL translation could be used without departing from the scope of the present invention. Any SQL query that establishes the selection criteria and

relationships among data will suffice. The intermediate representation itself is used as the tagger instructions 810. Those skilled in the art will recognize that a different instruction format could be used without departing from the scope of the present invention. Any instruction format that describes how to generate the marked up output from the results of the SQL query will suffice.

Query Execution

To produce the XML result, SQL query 800 is executed and its results are fed into tagger 240 of Xperanto 104. Tagger 240 uses tagger instructions 810 to generate the marked up output.

FIG. 9 is a flowchart that illustrates the method steps performed by the preferred embodiment of the present invention.

Block 900 represents the parsing of an XML query. Schema mapper 200 establishes a default XML schema (the Default View) for the data stored in RDBMS 102. Each table is mapped to a virtual XML document over which XML queries are issued. Parser 210 converts the XML query to a language-neutral intermediate representation, which is a sequence of operations that describe how the marked up output of the XML query is derived from the underlying relational tables.

Block 910 represents the rewriting of the intermediate to an equivalent form that may be easily translated into an SQL query. More specifically, operations to construct

and group XML elements 740 are reordered to appear last in the rewritten intermediate representation.

Block 920 represents the translation of the intermediate representation to: (1) SQL query 800 over the underlying relational table; and (2) tagging instructions 810 for tagger 240. Operations that select and relate data 730 are translated into SQL query 800, which establishes the selection criteria and relationships among data. Operations that construct and group XML elements 740 are translated into tagger instructions 810, which describe how to generate the marked up output from the results of SQL query 800.

Block 930 represents the execution of SQL query 800. The results of SQL query 800 are fed into tagger 240, which follows tagger instructions 810 to generate the marked up output.

In the foregoing description of the preferred embodiment, reference is made to the accompanying drawings that form a part hereof, and in which is shown by way of illustration a specific embodiment in which the invention may be practiced. It is to be understood that other embodiments may be utilized and that structural and functional changes or other modifications and variations may be made without departing from the scope of the present invention. For example, any type of computer, such as a mainframe, minicomputer, or personal computer, could be used to implement the present invention. In addition, the present invention is not limited by specific document formats or document query languages. For example, the present invention could also be

used with HTML or SGML. The description of the preferred embodiment has been presented for the purposes of illustration and description, and is not intended to be exhaustive or to limit the invention to the precise form disclosed. It is intended that the scope of the invention be limited not by this detailed description but rather by the claims appended hereto.

We claim:

1. A method of publishing relational data as XML, comprising the method steps of:

mapping a number of relational database tables to a number of virtual XML documents;

issuing XML queries over said virtual XML documents;

parsing said XML queries;

transforming said XML queries into a language-neutral intermediate representation;

rewriting said language-neutral intermediate representation into an equivalent form easily translated into an SQL query;

translating said equivalent form into an SQL query over said relational database tables and into tagging instructions passed to a tagger;

executing said SQL query to produce SQL query results passed to said tagger; and

generating XML output using said SQL query results and said tagging instructions.

2. The method of claim 1 wherein said method operates over a distributed computing network.

3. The method of claim 2 wherein said method operates over the Internet.

4. The method of claim 1 wherein said mapping step operates recursively.

5. The method of claim 1 wherein said mapping step operates manually.

6. The method of claim 1 wherein said mapping step maps said relational database tables to said virtual XML documents in a one-to-one manner.

7. The method of claim 1 wherein said language-neutral intermediate representation includes a sequence of operations describing:

how to select and relate data from said relational database tables; and

how to construct and group new XML elements from data bindings.

8. The method of claim 7 wherein said transforming step operates on at least one said relational database table and produces at least one output table.

9. The method of claim 7 wherein said operations include BIND operations.

10. The method of claim 7 wherein said operations include SELECT operations.

11. The method of claim 7 wherein said operations include CONSTRUCT operations.

12. The method of claim 7 wherein said operations include JOIN operations.

13. The method of claim 7 wherein said operations include GROUP operations.

14. The method of claim 7 wherein said operations include NEST operations.

15. The method of claim 1 wherein said rewriting step includes the further steps of:

eliminating both S and B whenever S is followed by a BIND operation B, where S denotes the sequence of CONSTRUCT, GROUP, and CONSTRUCT operations following a table access for a default view of a table T, leaving just the table access for T; and

replacing N by a JOIN operation, followed by S and a new GROUP operation which performs the child grouping that was previously done by N, where N denotes a NEST operation and S denotes any sequence of CONSTRUCT and GROUP operations for the child input of N.

16. The method of claim 1 wherein said rewriting step may operate repeatedly for deeper levels of nesting.

17. The method of claim1 wherein said tagger operates outside an RDBMS.

18. The method of claim 7 wherein said operations describing how to select and relate data are translated into an SQL query that establishes selection criteria and required relationships among data.

19. The method of claim 7 wherein said operations describing how to construct and group new XML elements are translated into said tagger instructions.

20. The method of claim 19 wherein said operations are reordered to be performed last.

21. The method of claim 19 wherein said language-neutral intermediate representation serves as said tagging instructions.

22. A system for publishing relational data as XML, comprising:

a schema mapper for mapping a number of relational database tables to a number of virtual XML documents;

an XML-QL engine for issuing XML queries over said virtual XML documents;

a parser for parsing said XML queries and for transforming said XML queries into a language-neutral intermediate representation;

a rewrite engine for rewriting said intermediate representation into an equivalent form easily translated into an SQL query;

a translator for translating said equivalent form into an SQL query over said relational database tables and into tagging instructions;

an RDBMS for executing said SQL query to produce SQL query results; and

a tagger for generating XML output using said SQL query results and said tagging instructions.

23. The system of claim 22 wherein said system operates over a distributed computing network.

24. The system of claim 23 wherein said system operates over the Internet.

25. The system of claim 22 wherein said schema mapper operates recursively.

26. The system of claim 22 wherein said schema mapper operates manually.

27. The system of claim 22 wherein said schema mapper maps said relational database tables to said virtual XML documents in a one-to-one manner.

28. The system of claim 22 wherein said language-neutral intermediate representation includes commands controlling how said system:
selects and relates data from said relational database tables; and.
constructs and groups new XML elements from data bindings.

29. The system of claim 28 wherein said parser operates on at least one said relational database table and produces at least one output table.

30. The system of claim 28 wherein said system performs BIND operations.

31. The system of claim 28 wherein said system performs SELECT operations.

32. The system of claim 28 wherein said system performs CONSTRUCT operations.

33. The system of claim 28 wherein said system performs JOIN operations.

34. The system of claim 28 wherein said system performs GROUP operations.

35. The system of claim 28 wherein said system performs NEST operations.

36. The system of claim 22 wherein said rewrite engine:

eliminates both S and B whenever S is followed by a BIND operation B, where S denotes the sequence of CONSTRUCT, GROUP, and CONSTRUCT operations following a table access for a default view of a table T, leaving just the table access for T; and

replaces N by a JOIN operation, followed by S and a new GROUP operation which performs the child grouping that was previously done by N, where N denotes a NEST operation and S denotes any sequence of CONSTRUCT and GROUP operations for the child input of N.

37. The system of claim 22 wherein said rewrite engine may operate repeatedly for deeper levels of nesting.

38. The system of claim 22 wherein said tagger operates outside an RDBMS.

39. The system of claim 28 wherein said system translates commands describing how to select and relate data into an SQL query that establishes selection criteria and required relationships among data.

40. The system of claim 28 wherein said system translates commands describing how to construct and group new XML elements into said tagger instructions.

41. The system of claim 40 wherein said commands are reordered to be performed last.

42. The system of claim 40 wherein said language-neutral intermediate representation serves as said tagging instructions.

43. A system for publishing relational data as XML, comprising:

means for mapping a number of relational database tables to a number of virtual XML documents;

means for issuing XML queries over said virtual XML documents;

means for parsing said XML queries and for transforming said XML queries into a language-neutral intermediate representation;

means for rewriting said intermediate representation into an equivalent form easily translated into an SQL query;

means for translating said equivalent form into an SQL query over said relational database tables and into tagging instructions;

means for executing said SQL query to produce SQL query results; and

means for generating XML output using said SQL query results and said tagging instructions.

44. A computer program product comprising a machine-readable medium including machine-executable instructions therein for publishing relational data as XML comprising the steps of:

mapping a number of relational database tables to a number of virtual XML documents;

issuing XML queries over said virtual XML documents;

parsing said XML queries;

transforming said XML queries into a language-neutral intermediate representation;

rewriting said language-neutral intermediate representation into an equivalent form easily translated into an SQL query;

translating said equivalent form into an SQL query over said relational database tables and into tagging instructions passed to a tagger;

executing said SQL query to produce SQL query results passed to said tagger; and

generating XML output using said SQL query results and said tagging instructions.

ABSTRACT

A method for publishing relational data as XML by translating XML queries into queries against an relational database. Conversion of the relational database into an XML database is not required. Each relational table is mapped to a virtual XML document, and XML queries are issued over these virtual documents. An XML query is parsed and transformed into a language-neutral intermediate representation, which is a sequence of operations describing how the output document is derived from the underlying relational tables. The intermediate representation is then translated into an SQL query over the underlying relational tables and into instructions for a tagger. The SQL query is executed, and the SQL query results are then fed into the tagger, which follows tagger instructions to generate the marked up output.

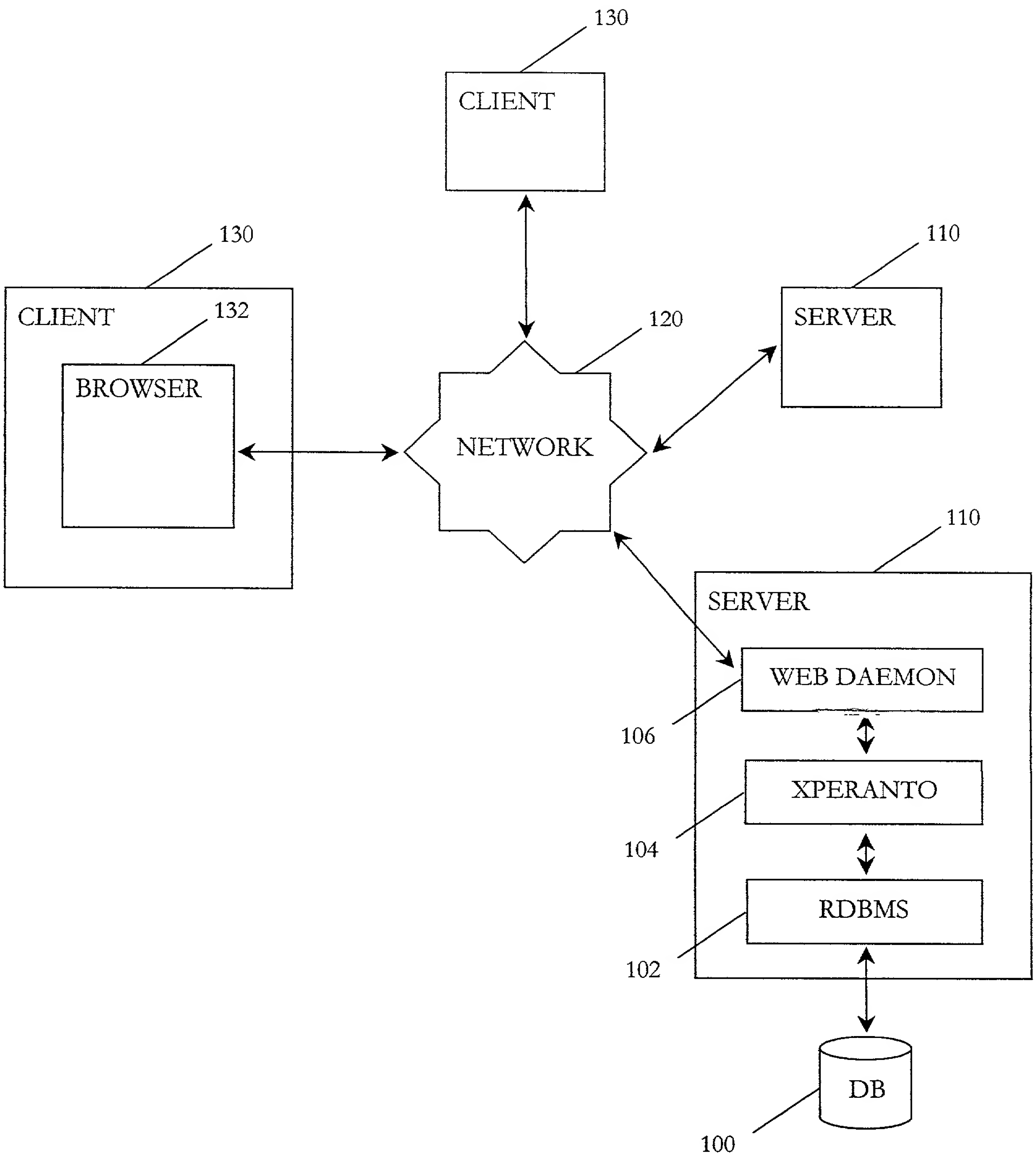
130
CLIENT
130
CLIENT
132
BROWSER
120
NETWORK
110
SERVER
110
SERVER
WEB DAEMON
106
XPERANTO
104
RDBMS
102
DB
100

FIG. 1

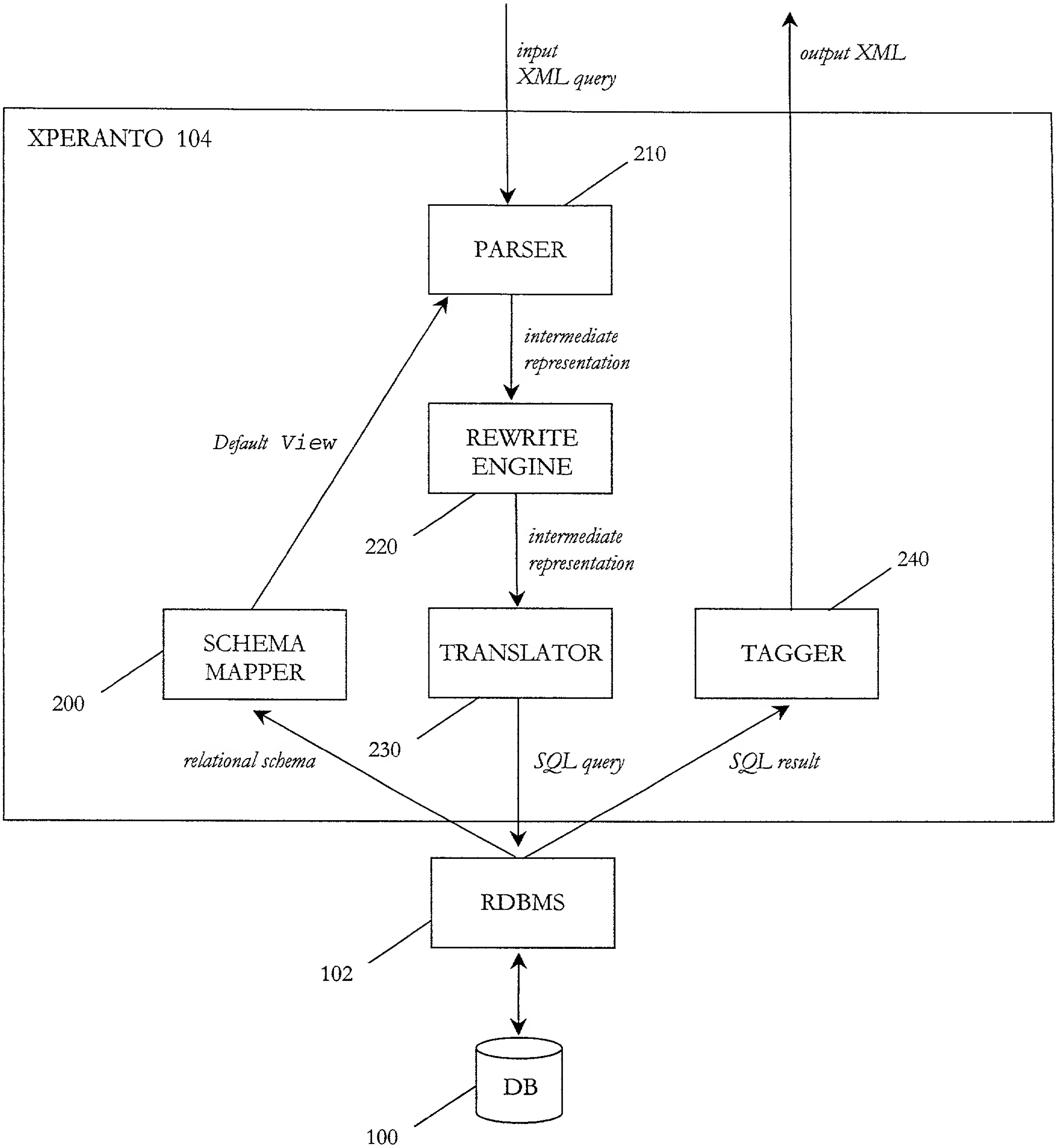
input
XML query
output XML
XPERANTO 104
210
PARSER
intermediate
representation
Default View
REWRITE
ENGINE
220
intermediate
representation
240
SCHEMA
MAPPER
TRANSLATOR
TAGGER
200
230
relational schema
SQL query
SQL result
RDBMS
102
DB
100

FIG. 2

DEPT TABLE 300

DNO	NAME	SIZE
1	Finance	100
2	Marketing	200
3	Sales	50
...		

EMP TABLE 310

DNO	ENO	NAME	AGE
1	1	J. Skolem	50
2	2	L. Hubbard	33
...			

Default XML View of the DEPT Table 320

```
<DEPT>
  <ROW>
    <DNO> 1 </DNO> <NAME> Finance </NAME> <SIZE> 100 </SIZE>
  </ROW>
  <ROW>
    <DNO> 2 </DNO> <NAME> Marketing </NAME> <SIZE> 200 </SIZE>
  </ROW>
  <ROW>
    <DNO> 3 </DNO> <NAME> Sales </NAME> <SIZE> 50 </SIZE>
  </ROW>
  ...
</DEPT>
```

Default XML View of the EMP Table 330

```
<EMP>
  <ROW>
    <DNO> 1 </DNO> <ENO> 1 </ENO> <NAME> J. Skolem </NAME> <AGE> 50 </AGE>
  </ROW>
  <ROW>
    <DNO> 2 </DNO> <ENO> 2 </ENO> <NAME> L. Hubbard </NAME> <AGE> 33 </AGE>
  </ROW>
  ...
</EMP>
```

FIG. 3

Query Over the Default View

```
WHERE
  <DEPT>
    <ROW>
      <DNO> $DNO </DNO> <NAME> $DNAME </NAME>
      <SIZE> $DSIZE </SIZE>
    </ROW>
  </DEPT> IN DefaultView,
  $DSIZE > 75
CONSTRUCT
  <BIGDEPT>
    <NAME> $DNAME </NAME>
    <SIZE> $DSIZE </SIZE>
    {WHERE
       <EMP>
         <ROW> <DNO> $DNO </DNO> <NAME> $ENAME </NAME> </ROW>
       </EMP> IN DefaultView
     CONSTRUCT
       <EMP> <NAME> $ENAME </NAME> </EMP>}
  </BIGDEPT>
```

XML Produced by the Query

```
<BIGDEPT>
  <NAME> Finance </NAME>
  <SIZE> 100 </SIZE >
  <EMP> <NAME> J. Skolem </NAME> </EMP>
  ...
</BIGDEPT>
<BIGDEPT>
  <NAME> Marketing </NAME>
  <SIZE> 200 </SIZE >
  <EMP> <NAME> L. Hubbard </NAME> </EMP>
  ...
</BIGDEPT>
...
```

FIG. 4

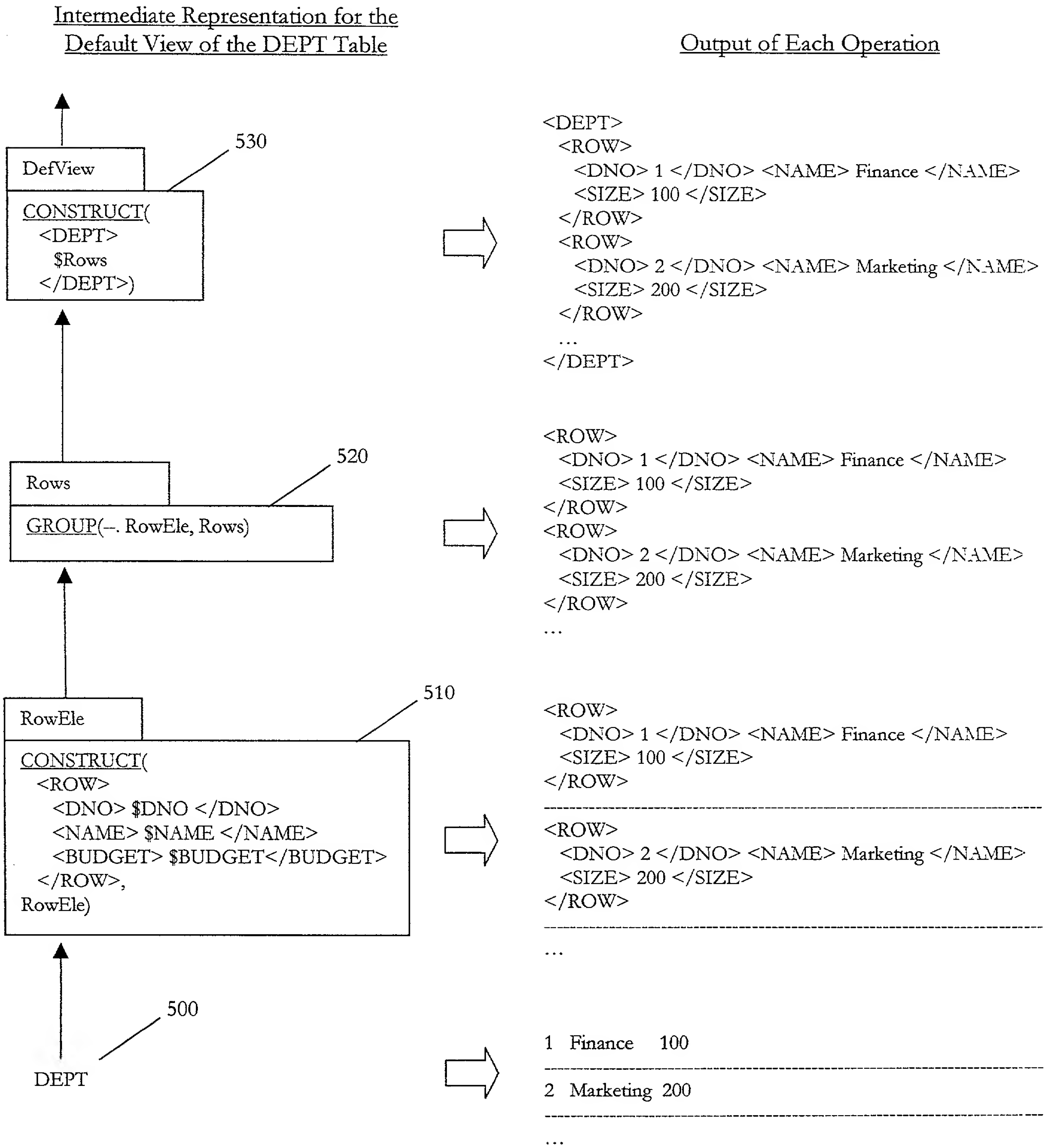
Intermediate Representation for the Default View of the DEPT Table
Output of Each Operation
530
DefView
CONSTRUCT(
<DEPT>
$Rows
</DEPT>)
<DEPT>
<ROW>
<DNO> 1 </DNO> <NAME> Finance </NAME>
<SIZE> 100 </SIZE>
</ROW>
<ROW>
<DNO> 2 </DNO> <NAME> Marketing </NAME>
<SIZE> 200 </SIZE>
</ROW>
...
</DEPT>
520
Rows
GROUP(–. RowEle, Rows)
<ROW>
<DNO> 1 </DNO> <NAME> Finance </NAME>
<SIZE> 100 </SIZE>
</ROW>
<ROW>
<DNO> 2 </DNO> <NAME> Marketing </NAME>
<SIZE> 200 </SIZE>
</ROW>
...
510
RowEle
CONSTRUCT(
<ROW>
<DNO> $DNO </DNO>
<NAME> $NAME </NAME>
<BUDGET> $BUDGET</BUDGET>
</ROW>,
RowEle)
<ROW>
<DNO> 1 </DNO> <NAME> Finance </NAME>
<SIZE> 100 </SIZE>
</ROW>
<ROW>
<DNO> 2 </DNO> <NAME> Marketing </NAME>
<SIZE> 200 </SIZE>
</ROW>
...
500
DEPT
1 Finance 100
2 Marketing 200
...

FIG. 5

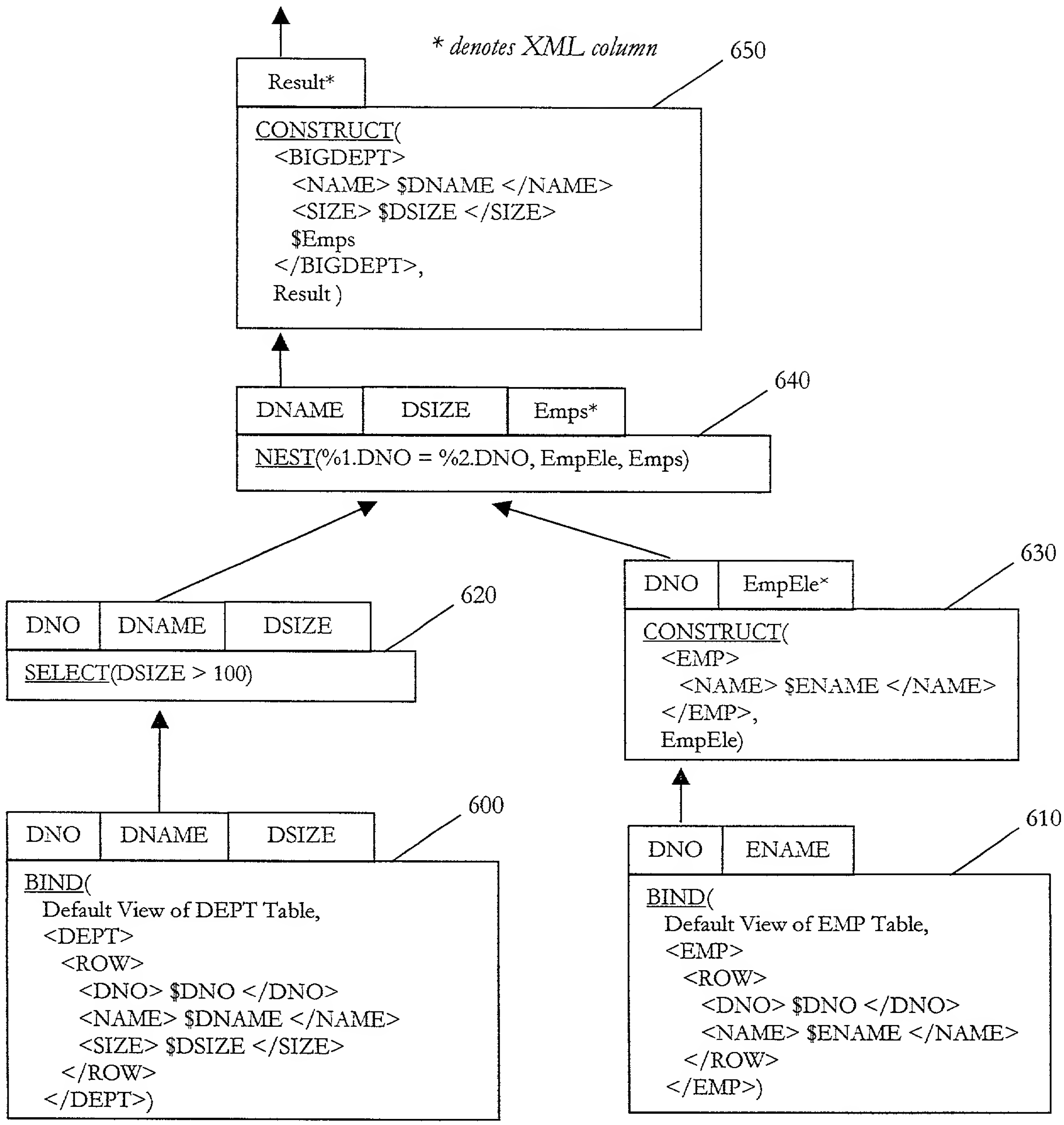
* denotes XML column
650
Result*
CONSTRUCT(
<BIGDEPT>
<NAME> $DNAME </NAME>
<SIZE> $DSIZE </SIZE>
$Emps
</BIGDEPT>,
Result)
640
DNAME
DSIZE
Emps*
NEST(%1.DNO = %2.DNO, EmpEle, Emps)
630
DNO
EmpEle*
CONSTRUCT(
<EMP>
<NAME> $ENAME </NAME>
</EMP>,
EmpEle)
620
DNO
DNAME
DSIZE
SELECT(DSIZE > 100)
600
DNO
DNAME
DSIZE
BIND(
Default View of DEPT Table,
<DEPT>
<ROW>
<DNO> $DNO </DNO>
<NAME> $DNAME </NAME>
<SIZE> $DSIZE </SIZE>
</ROW>
</DEPT>)
610
DNO
ENAME
BIND(
Default View of EMP Table,
<EMP>
<ROW>
<DNO> $DNO </DNO>
<NAME> $ENAME </NAME>
</ROW>
</EMP>)

FIG. 6

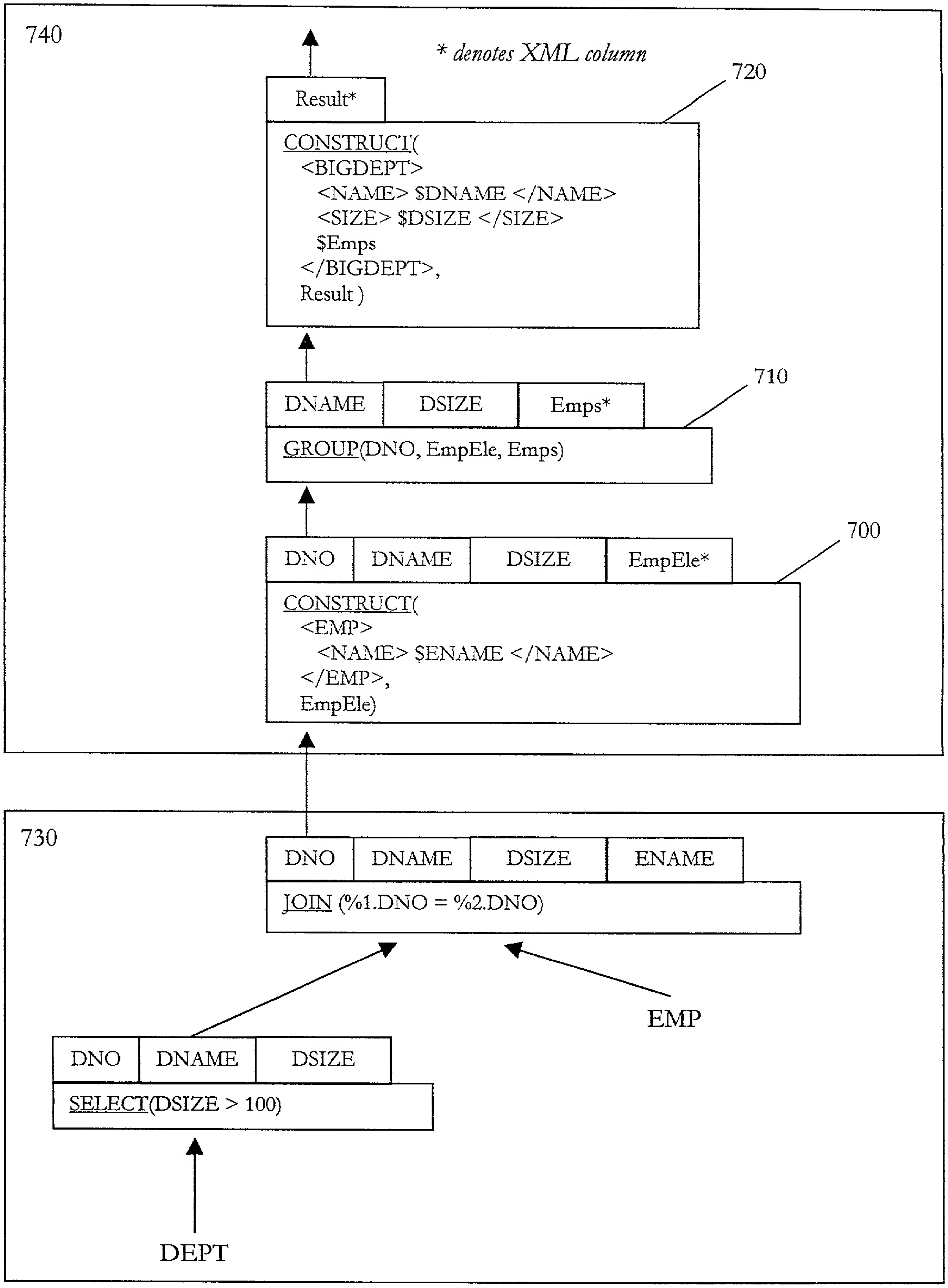
740
* denotes XML column
720
Result*
CONSTRUCT(
<BIGDEPT>
<NAME> $DNAME </NAME>
<SIZE> $DSIZE </SIZE>
$Emps
</BIGDEPT>,
Result)
710
DNAME
DSIZE
Emps*
GROUP(DNO, EmpEle, Emps)
700
DNO
DNAME
DSIZE
EmpEle*
CONSTRUCT(
<EMP>
<NAME> $ENAME </NAME>
</EMP>,
EmpEle)
730
DNO
DNAME
DSIZE
ENAME
JOIN (%1.DNO = %2.DNO)
EMP
DNO
DNAME
DSIZE
SELECT(DSIZE > 100)
DEPT

FIG. 7

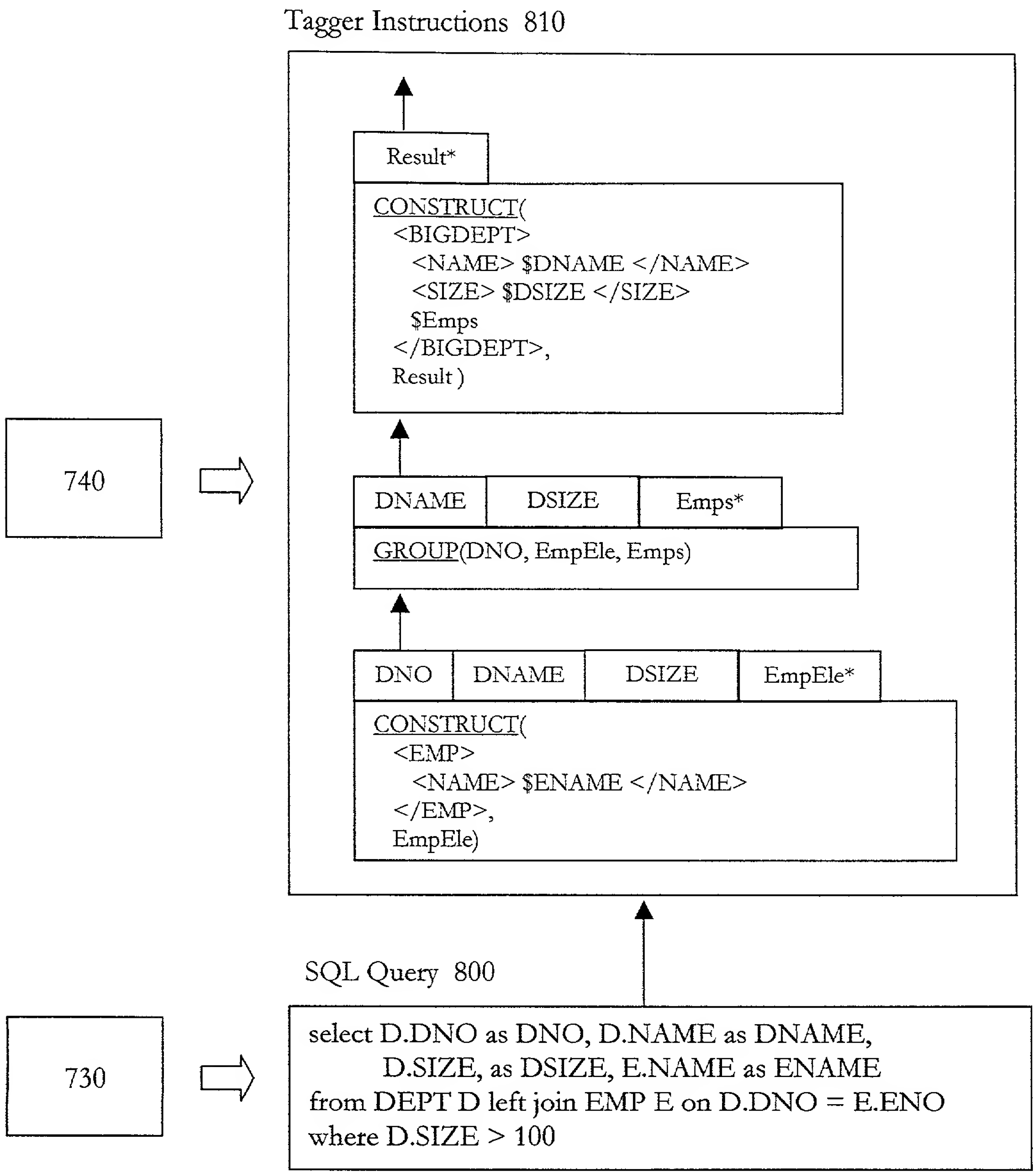
Tagger Instructions 810
Result*
CONSTRUCT(
<BIGDEPT>
<NAME> $DNAME </NAME>
<SIZE> $DSIZE </SIZE>
$Emps
</BIGDEPT>,
Result)
740
DNAME
DSIZE
Emps*
GROUP(DNO, EmpEle, Emps)
DNO
DNAME
DSIZE
EmpEle*
CONSTRUCT(
<EMP>
<NAME> $ENAME </NAME>
</EMP>,
EmpEle)
SQL Query 800
730
select D.DNO as DNO, D.NAME as DNAME,
D.SIZE, as DSIZE, E.NAME as ENAME
from DEPT D left join EMP E on D.DNO = E.ENO
where D.SIZE > 100

FIG. 8

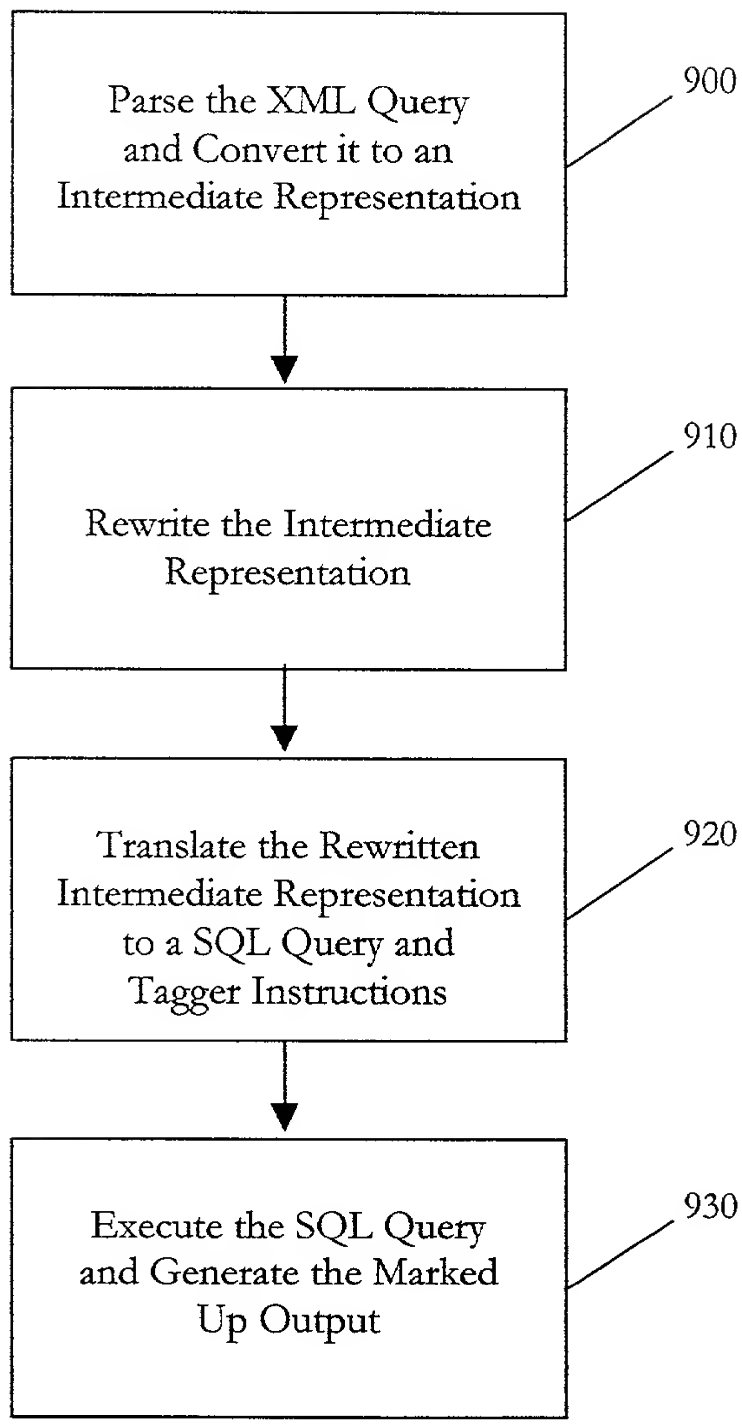
Parse the XML Query and Convert it to an Intermediate Representation
900
Rewrite the Intermediate Representation
910
Translate the Rewritten Intermediate Representation to a SQL Query and Tagger Instructions
920
Execute the SQL Query and Generate the Marked Up Output
930

FIG. 9

 Docket No. AM9-99-0247

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

USING AN XML QUERY LANGUAGE TO PUBLISH RELATIONAL DATA AS XML

the specification of which is attached hereto unless the following box is checked:

was filed on ________________
as United States Application Number or PCT International Application Number ________________
and was amended on ________________ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR §1.56.

I hereby claim foreign priority benefits under 35 USC §119(a-d) or §365(b) of any foreign application(s) for patent or inventor's certificate, or §365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or PCT International application having a filing date before that of the application on which priority is claimed.

Prior Foreign Application(s): Priority Not Claimed

(Number)	(Country)	(Day/Month/Year Filed)

I hereby claim the benefit under 35 USC §119(e) of any United States provisional application(s) listed below:

Provisional Application(s):

(Application Number)	(Filing Date)

I hereby claim the benefit under 35 USC §120 of any United States application(s), or §365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of 35 USC §112, I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR §1.56 which became available between the filing date of the prior application and the national or PCT International filing date of this application.

(Application Number)	(Filing Date)	(Status - patented, pending, abandoned)

Power of Attorney:

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

Richard M. Ludwin	(#33,010)	Thomas R. Berthold	(#28,689)
Khanh Q. Tran	(#41,352)	Marc D. McSwain	(#44,929)
Alison D. Mortinger	(#39,306)		

 Docket No. AM9-99-0247

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

Address all telephone calls to:

Marc D. McSwain

(408) 927-3364

Address all correspondence to:

Marc D. McSwain
IBM Corporation
Intellectual Property Law (C4TA/J2B)
650 Harry Road
San Jose, CA 95120-6099

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of first inventor: Michael James Carey

Inventor's signature: Date: 3/17/2000

Residence: 1473 Almaden Valley Drive, San Jose, California 95120

Citizenship: United States of America

Full name of second inventor: Jayavel Shanmugasundaram

Inventor's signature: Date: 20 MAR 2000

Residence: 7147 Via Maria, San Jose, California 95139

Citizenship: India

 Docket No. AM9-99-0247

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

//

Full name of third inventor: Eugene Jon Shekita

Inventor's signature: Eugene Jon Shekita Date: 3/20/00

Residence: 6599 Winterset Way, San Jose, California 95120

Citizenship: United States of America

//

Full name of fourth inventor: Narayana Iyer

Inventor's signature: Date: 3/20/00

Residence: 4300 The Woods Drive Apt 1522, San Jose, California 95136

Citizenship: India

//